IoT S&S:
Embedded Systems

Gabe Parmer

IoT Project Brainstorming

- Project goal is some combination of:
  - Build something **cool**
  - Build an exciting demo
  - Learn about
    - Embedded system programming
    - Interacting with the world
      - Note: this takes longer than you'd think
    - Security in IoT
    - Cloud programming

IoT Project Brainstorming

- Project goal is some combination of:
  - Build something **cool**
  - Build an exciting demo
  - Learn about
    - Embedded system program
    - Interacting with the world
      - Note: this takes longer than you'd think
    - Security in IoT
    - Cloud programming

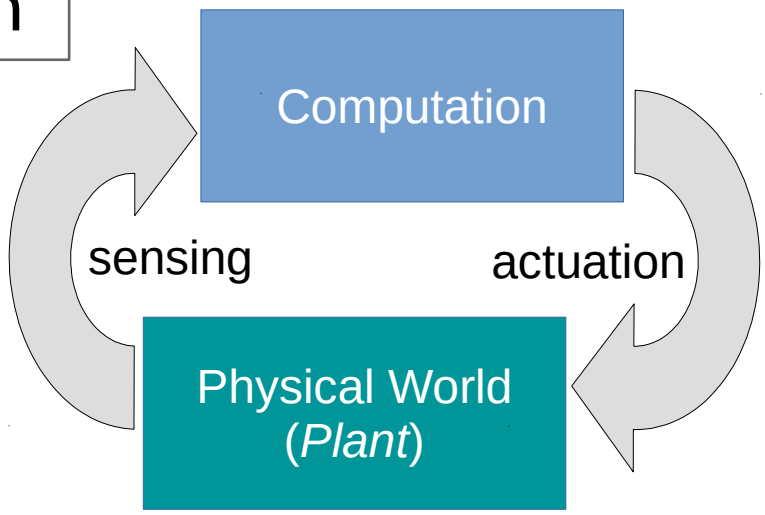
Department has kindly provided $$$

IoT Project Criteria

- See criteria online
  - Everything is up for discussion
- Group of 3 by default
  - 1 embedded system, 1 cloud, 1 sensors/actuators
  - 2 embedded, 1 cloud
  - 2 cloud, 1 embedded
  - 3 embedded, 3 cloud
  - ...

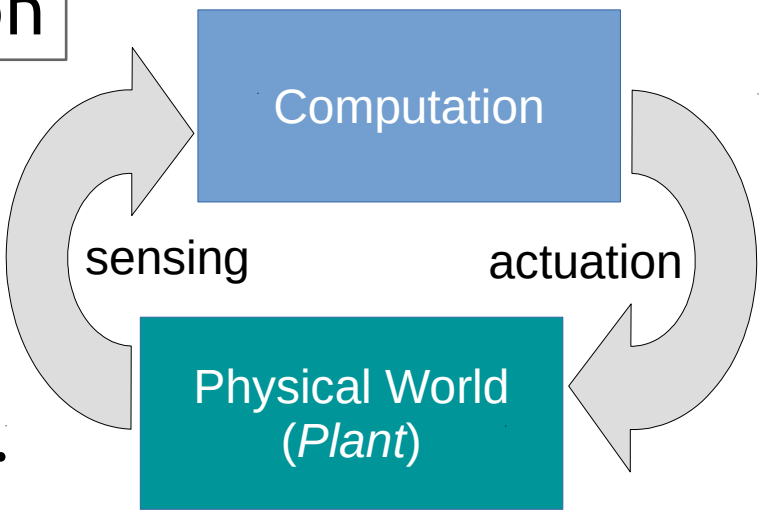
Embedded Systems

- Computers that interact with physical world
  - Sensors + actuators + computation
- Replace mechanical w/ digital
  - Microwaves, toys
  - Cars, airplanes



Embedded Systems

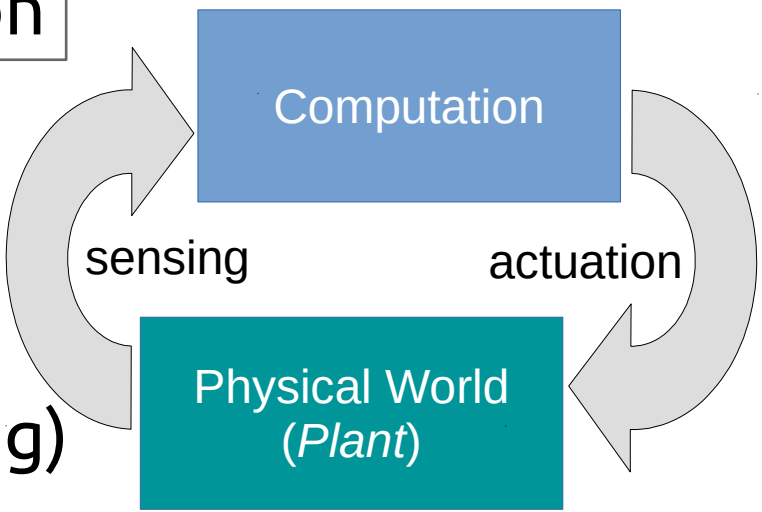
- Computers that interact with physical world
  - Sensors + actuators + computation



- Sensors
  - Visual/audio: Camera/mic
  - Environment: temp, barometer, ...
  - Position: IMU/Gyro
  - Range: LIDAR, infrared/ultrasonic

Embedded Systems

- Computers that interact with physical world
  - Sensors + actuators + computation



- Acuators
  - DC motors (high RPM)
  - Servo/Stepper motors (positioning)
  - Human: Screens/speakers
  - Environment: lights, HVAC

Embedded System Environments

APIs and interfaces for

- Managing computational resources
  - threads, mutexes, memory allocation
- Managing physical resources
  - GPIO
  - Voltage control pins
- Wireless

Embedded System Environments

APIs and interfaces for

- Managing computational resources
  - threads, mutexes, memory allocation
- Managing physical resources
  - GPIO
  - Voltage control pins
- Wireless

Embedded System Environments II

Three classes of interfaces

- Arduino:
  - simple, single control loop, limited multitasking
- RTOS + microcontroller:
  - time-sensitive, cheap, low power, versatile
- OS + GP microprocessor:
  - your phone in a Single Board Computer (SBC)

Embedded Syst

Three classes of interf

- Arduino:
  - simple, single contro
- RTOS + microcontro
  - time-sensitive, chea
- OS + GP microproce
  - your phone in a Sing

- Large number of libraries
- Loop with timing driven by *delay(…)*
- Example: one sensor that you can read every 1 millisecond

```
void loop() {
    delay(1);
    reading = read(DEV1);
    process(reading);
}
```

- What if *2 devices*: read every 2ms, and every 5ms. **Discuss**
- What if a devices can be read every 1 ms, and another requires 50ms processing? **Discuss**
- *By default*: I'd like you to avoid Arduino unless other aspects of your system are quite difficult

Embedded System Environments II

Three classes of i

- Arduino:
  - simple, single
- RTOS + mi
  - time-sensitive
- OS + GP microp
  - your phone in

RTOSes: multiple flows of control (e.g. one per sensor)

- What if 2 devices: read every 2ms, and every 5ms.

```
periodic_wait(2000);
while(1) {
    r = read(DEV1);
    process(r);
    wait();
}
```

```
periodic_wait(5000);
while(1) {
    r = read(DEV2);
    process(r);
    wait();
}
```

- Coordination between these “threads”: mutexes/semaphores/channels/priority
- How might you choose a thread’s priority? **Discuss**

Embedded System Environments II

Three classes of interfaces

- Arduino:
  - simple, single control loop, limited multitasking
- RTOS + microcontroller
  - time-sensitive, cheap, low p
- OS + GP microprocesso
  - your phone in a Single Boa

Microcontroller vs. GP microprocessor?

- What's the difference?
- Examples of domains where you'd see one versus the other?
- **Discuss!**

Embedded System I

Three classes of interfaces

- Arduino:
  - simple, single control loop
- RTOS + microcontroll
  - time-sensitive, cheap, low
- OS + GP microproces
  - your phone in a Single Boa

Microcontroller

- Power
- Size
- Memory:
  - 16KB-1MB SRAM
  - 32KB-8MB Flash
- Compute:
  - 8-800 MhZ
- Zero or MPU-based protection
- Cheap!
- Example: STM Arm-M4

GP microprocessor

- Think: your cell phone
- Power: -ish
- Virtual memory protection
- Examples:
  raspberry Pi, beaglebone

Embedded System Environments II

Three classes of interfaces

- Arduino:
  - simple, single control loop, limited multitasking
- RTOS + microcontroller:
  - time-sensitive, cheap, low p
- OS + GP microprocesso
  - your phone in a Single Boa

GP OS: Linux

...that's it...

Timeline

See Syllabus:

- Team selection: *delay(1week); select([members]);*
- Project plan w/ security analysis:
  delay(2week); select(project);
- Milestones & peer eval
- Final presentation/demo
- Public demo

Timeline

See Syllabus:

- Discussion leader
  - 2 weeks before:
    - Post *pdf* link
    - Create issue for paper on github for discussion
  - the week before: send Gabe presentation + attend OOs
  - 1 day before: summarize discussion
  - Day of: present ~20 minutes

Timeline

See Syllabus:

- Non-discussion leader
  - Choose paper for comprehension review (on T or R)
  - Choose paper for critical review (on the other of T/R)
  - 2 days before: post reviews on github issue
  - Come prepared to discuss!

Project Ideas/Inspiration

See class repo...